APPLICATION FOR UNITED STATES LETTERS PATENT

INVENTORS: Kushagra VAID
San Jose, California

Suresh MARISETTY
San Jose, California

Yaron SHRAGAI
Natick, Massachusetts

Koichi YAMADA
Los Gatos, California

Rajendra KURAMKOTE
Newcastle, Washington

Scott BRENDEN
Bothell, Washington

TITLE: POISONED ERROR SIGNALING FOR PROACTIVE OS RECOVERY

ASSIGNEE: Intel Corporation
Santa Clara, California

ATTORNEYS/ AGENTS: Venable, LLP
Box 34385
Washington, DC 20043-9998
Telephone: (202) 962-4800
Facsimile: (202) 962-8300

ATTORNEY DOCKET NO.: 42339-192083

FIELD OF THE INVENTION

[0001] Embodiments of the present invention may relate to the field of fault-tolerant computing and, more specifically, to the detection and mitigation of the effects of data errors.

BACKGROUND OF THE INVENTION

[0002] Despite the presence of error-control coding (ECC) in computer systems, it is still possible for uncorrectable errors to occur. For example, in many systems a two-bit ECC (2xecc) error may not be correctable. If data containing such errors is consumed by the processor, it may cause spurious computational results, or it may even cause the operating system (OS) to go down, e.g., by means of a machine check abort (MCA).

[0003] One way of dealing with such uncorrectable errors is, upon detection of such errors by the processor, to assert a global MCA. This has the effect of bringing down the system, however. As a result, the availability and reliability of the computer system are reduced.

[0004] One refinement of this process is to detect uncorrectable data errors and to mark the data containing such errors. This technique is known as "data poisoning." As a result, if the processor detects that the data it is about to consume has been "poisoned," it can invoke an MCA to avoid the consumption of the poisoned data. While this provides a more convenient technique by which a processor can detect the presence of such uncorrectable errors, it provides only an incremental improvement in availability and reliability, as it is still necessary to bring down the system.

DEFINITIONS

[0005] Components/terminology used herein for one or more embodiments of the invention is described below:

[0006] In some embodiments, "computer" may refer to any apparatus that is capable of accepting a structured input, processing the structured input according to prescribed rules, and producing results of the processing as output. Examples of a computer may include: a computer; a general purpose computer; a supercomputer; a mainframe; a super mini-computer; a mini-computer; a workstation; a microcomputer; a server; an interactive television; a hybrid combination of a computer and an interactive television; and application-specific hardware to emulate a computer and/or software. A computer may have a single processor or multiple processors, which may operate in parallel and/or not in parallel. A computer may also refer to two or more computers connected together via a network for transmitting or receiving information between the computers. An example of such a computer may include a distributed computer system for processing information via computers linked by a network.

[0007] In some embodiments, a "machine-accessible medium" may refer to any storage device used for storing data accessible by a computer. Examples of a machine-accessible medium may include: a magnetic hard disk; a floppy disk; an optical disk, like a CD-ROM or a DVD; a magnetic tape; a memory chip; and a carrier wave used to carry machine-accessible electronic data, such as those used in transmitting and receiving e-mail or in accessing a network.

[0008] In some embodiments, "software" may refer to prescribed rules to operate a computer. Examples of software may include: code segments; instructions; computer programs; and programmed logic.

[0009] In some embodiments, a "computer system" may refer to a system having a computer, where the computer may comprise a computer-readable medium embodying software to operate the computer.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Preferred embodiments of the invention will now be described in connection with the associated drawings, in which:

[0011] Figure 1 depicts a conceptual block diagram of a computer system adapted to implement an embodiment of the invention; and

[0012] Figure 2 depicts a flowchart of a method implementing an embodiment of the invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0013] Figure 1 shows a conceptual block diagram of a computer system adapted to implement an embodiment of the present invention. The computer system may include at least one processor 101, accompanied by system memory 100, used to store system software for running on processor 101. Processor 101 may also have an associated processor cache 102, typically used to store data to be processed by an application being run on processor 101. Additional memory 104 may be available to store data, software, etc. In some embodiments, memory 104 and system memory 100 may comprise

locations within a common physical memory device. Additionally, in some embodiments, there may be multiple memory devices comprising system memory 100, memory 104, and/or processor cache 102.

[0014] As shown, system memory 100 may be coupled to processor 101 through a bus 107, and processor 101 may also be coupled to processor cache 102 through a bus 108. While shown directly coupled in Figure 1, it would be understood by one skilled in the art that the processor and all of its memories may be inter-linked by one or more buses.

[0015] Figure 1 shows a bus 105 that links various components of the computer system. For example, data from one component (e.g., memory 104) may be sent to processor 101 for processing or for storing in system cache 102 through bus 105. Furthermore, bus 105 may include multiple buses.

[0016] Also shown in Figure 1 is an error-control decoder 103. Data in the computer system may be protected using error-control coding. Error-control decoder 103 may contain logic for checking data to ensure that it is error-free and for correcting any errors that may be correctable using the error-control coding. Error-control decoder 103 may be implemented in hardware or as software/firmware, run in conjunction with processor 101 (i.e., implemented on or as part of processor 101), or as a combination of both. Error-control decoder 103 may further be implemented as part of one or more of the memory blocks (100, 102, 104).

[0017] Furthermore, the system may include multiple error-control decoders 103, implemented in any of the ways previously described, and this may enable the system to check for errors in multiple locations. For example, one error-control decoder 103 may be adapted to check for outbound data errors from processor 101 to a memory unit (e.g.,

memory 104), while a second error-control decoder 103 may be adapted to check for inbound errors. In the case in which an error-control decoder 103 checks for outbound errors, there may be further capability, e.g., within the memory unit, to inform an operating system of the computer system of the presence of corrupted data, in a similar fashion to the process described below.

[0018] Error-control decoder 103 may be further adapted to mark data as being "bad," i.e., poisoning the data, if it determines that the data contains one or more uncorrectable errors. Typically, error-control decoder 103 may poison a larger unit of data, for example, a cache line, that contains the erroneous data; however, the invention is not limited to this case. Error-control decoder 103 may, for example, operate on data in a memory unit, typically, processor cache 102, or data being transmitted over a bus 105, 107, or 108.

[0019] The computer system of Figure 1 may further include control logic 106. Control logic 106 may be used when data is poisoned by error-control decoder 103 to interact with processor 101 and to control the transmission of notification of data poisoning to processor 101 (i.e., to an operating system running on the processor). Control logic 106 may be implemented as part of processor 101, in addition to the implementation shown in Figure 1. Such data poisoning notification may include information that may enable recovery of the poisoned data. For example, such information may include a target address corresponding to a the data in which the errors occurred.

[0020] In a particular embodiment of the invention, a handshaking process may occur between the control logic 106 and an operating system running on processor 101, in which recovery-related information is passed to the operating system.

[0021] Figure 2 is a flowchart depicting an embodiment of the inventive method. Overall, there may be operations that are executed in hardware and/or firmware and/or non-OS software 212 and those that may be executed by an operating system (OS) 213 running on processor 101. In particular, the components included in 213 may be included in an OS MCA routine.

[0022] First, data units may be checked for errors 200; this may involve error-control decoder 103. It may then be determined if errors that are detected are correctable or uncorrectable 201. If they are correctable, they may be corrected 214 and may then be reported to the OS for logging 202 (as being correctable). In this case, no further action may be necessary with respect to that data unit.

[0023] On the other hand, if the error-control decoder 103 determines that there are uncorrectable errors 201, then the system may determine if the resulting uncorrectable errors cause a data poisoning (DP) error 203. In particular, a system may be designed so that DP errors correspond to resident data units found in certain storage locations (e.g., cache 102, or memory 100 or 104) or during its transmission to or from these storage locations over a bus (e.g., bus 105, 107, or 108). Alternatively, the system may be designed to check for errors in data units in other portions of the system. In summary, the system designer may decide what uncorrectable errors are designated as being DP errors and which, if any, are not.

[0024] If an uncorrectable error situation arises that is not a DP error, the corresponding data unit, including the erroneous data, may have been consumed (i.e., by an application running on the computer system or by the operating system); the process, accordingly, may skip to block 206 (when that data unit is consumed).

[0025] If an uncorrectable error situation corresponds to a DP error, then a DP flag associated with the corresponding data unit may be set 204. If the system has a policy for addressing DP errors 205 in which immediate notification (action) occurs, the process may continue to block 206. Otherwise, the error may only be reported for logging 202. Note that whether the system provides immediate notification is an option that may be determined by a system designer or by a system operator, depending upon the particular system.

[0026] In the case where the process has progressed to block 206, the process has entered the portion that may be performed by the OS. When reading in a data unit, the OS may determine if the DP flag has been set 206. If yes, then the data unit was poisoned and may be removed 207. System operation may then resume 211 without experiencing the results of consuming the erroneous data.

[0027] If the DP flag has not been set, however, whether or not the data errors can be mitigated may depend on where the erroneous data lies 208. If the data unit containing the erroneous data is in user space, the OS may still detect the presence of the error, terminate the application, and remove the data unit 210. In this case, system operation may resume 211, as above.

[0028] On the other hand, if the erroneous data is in OS space, the erroneous data may be consumed by the OS 209. As a result, whether or not the OS is able to recover may depend on in what location the error(s) occurred. If the OS can not recover, it may initiate a graceful shutdown, which may still provide some degree of error mitigation, as long as the effects of the error(s) are not permanent (e.g., where permanently-stored data has not been corrupted as a result of the error(s)).

[0029] In a further variation on the above method and computer system, it may be possible for either the operating system or the system designer to select whether or not to invoke a data poisoning method as described above. Implementing such a feature may permit the system designer or operator to select an error-handling technique in accordance with system requirements. For example, there may be a software-visible control bit that permits the selection of either immediate notification of an uncorrectable error situation (i.e., invoking a policy of addressing data poisoning; see Figure 2, reference numeral 205) or to defer notification (i.e., to let the OS detect if data contains uncorrected errors). If so, the detection of whether this control bit has been set may occur in block 205 of Figure 2.

[0030] The invention has been described in detail with respect to preferred embodiments, and it will now be apparent from the foregoing to those skilled in the art that changes and modifications may be made without departing from the invention in its broader aspects. The invention, therefore, as defined in the appended claims, is intended to cover all such changes and modifications as fall within the true spirit of the invention.